NASA Technical Memorandum 84648

NASA-TM-84648 19860004780

FLEXWAL: A COMPUTER PROGRAM FOR PREDICTING THE WALL MODIFICATIONS FOR TWO-DIMENSIONAL, SOLID, ADAPTIVE-WALL WIND TUNNELS

Joel L. Everhart

November 1983

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

4 1 1 RN/NASA-TM-84648
DISPLAY 04/2/1
84X10066*# ISSUE 2 CATEGORY 2 RPT#: NASA-TM-84648 NAS 1.15:84648
CNT#: PROJ. FEDD 83/11/00 54 PAGES UNCLASSIFIED DOCUMENT DOMESTIC
UTTL: FLEXWAL: A computer program for predicting the wall modification for two-dimensional, solid, adaptive-wall wind tunnels TLSP: An Early Domestic Dissemination Report
AUTH: A/EVERHART, J. L.
CORP: National Aeronautics and Space Administration. Langley Research Center, Hampton, Va. SAP: Avail: NASA Industrial Applications Centers only to U.S. requesters
MAJS: /*AERODYNAMIC INTERFERENCE/*TRANSONIC FLOW/*WIND TUNNEL WALLS
MINS: / CAUCHY INTEGRAL FORMULA/ SUBSONIC FLOW/ TRANSONIC FLOW
ABA: Author
ABS: A program called FLEXWAL for calculating wall modifications for solid, adaptive-wall wind tunnels is presented. The method used is the iterative technique of NASA TP-2081 and is applicable to subsonic and transonic test conditions. The program usage, program listing, and a sample case are given.

ENTER:

SUMMARY

A program called FLEXWAL is presented. The program was written to predict the wall modifications necessary to remove the upper and lower wall interference effects in solid, flexible, adaptive-wall wind tunnels. The procedure used is the iterative method of NASA TP-2081 and is valid for both subsonic and transonic test conditions. Instructions on program usage, functional statements of program routines, a program listing, and a sample case are given.

INTRODUCTION

The computer program FLEXWAL described in this report is used to calculate the required upper and lower boundary modifications necessary for removing wind-tunnel wall interference effects in solid flexible-wall wind tunnels. The method used is the iterative technique of reference 1 and requires experimentally obtained information as input. The program is completely self-contained and is written in a modular form so that changes to the program for specific applications or wind-tunnel facilities can be made with ease.

Both analytical and experimental applications of the program at subsonic and transonic speeds have been previously presented along with the convergence properties of the method (See ref. 1.). This report gives a general description of the method, instructions for using the program, sample input, functional statements of routines used, and a listing of the computer code.

GENERAL DESCRIPTION

When testing in a wind tunnel, the flow field about the model is constrained to something different than that which would be present in free air. The principal difference is referred to as the wall interference and is generally taken as a perturbation about the free-air case. Classical methods for determining the wind-tunnel wall interference make mathematical approximations of the perturbation from which correction formulae are derived. The present method is that of reference 1 and uses the Cauchy integral formula to analytically extend the real flow field in the wind tunnel to infinity by solving for an imaginary flow exterior to the wind tunnel. These two flows are then iteratively coupled at the boundary with continuity being established when the measured data match the results of calculated data at the wall. If there is a mismatch then the method predicts a correction to the wind-tunnel wall, new data are measured, and the continuity is again checked.

Functional statements of the routines used in the FLEXWAL program are given in Appendix A and a listing of the program is contained in Appendix B. The program is completely self-contained and modular in construction. All input data are read into subroutine INPT and an echo listing of the input is given in the output listing. Subroutine SCALE uses the Goethert scaling laws to remove the effects of compressibility at the test Mach number prior to application of the correction technique and then, after completion of the correction calculations, SCALE is called again so as to include the compressibility at the desired Mach number MSET. MSET

has been included for consistency during the iterative process because often the desired Mach number is slightly different than that recorded during the test due to experimental error. Subroutine SMOOTH is a simple least squares smoothing subroutine which may or may not be needed depending on the quality of the experimentally obtained data. For each set of five adjacent pressures, a least squares parabola is determined and the center pressure is calculated from the least squares solution. When all pressures have been smoothed, a single pass has been completed and the process will be repeated as specified by the IS parameter in the input. The subroutine WALLDTA determines values of the perturbation velocities and sets up an integration contour as specified in reference 1. A spline through the wall jack coordinates allows the wallposition to be determined at intermediate stations and by taking the derivative of the spline curve, the wall slopes (i.e. vertical velocities) may also be obtained along the walls. In the present form of the computer code, the vertical velocity components on the upstream and downstream legs of the contour have been set equal to the average of those on the upper and lower walls at the same station. This can easily be changed if the appropriate data are measured or better approximated.

The actual application of the method is in subroutine VJACK. It is here that the contour integration is performed for the velocities at the wall jack stations. It is important to note that

the most upstream and downstream wall jacks do not actually have to lie on the data contour. The program will determine whether the velocity is being calculated at a field point or a point on the contour and will adjust the solution accordingly. The difference is a factor of 2 which can be seen by referring to reference 1. Subroutine DELYJ calculates the predicted wall change by integrating slopes determined from the new jack velocities along each wall. Convergence of the method is declared when these changes become sufficiently small or when some other criteria such as negligible variations in the airfoil force coefficients is met. Subroutine NEWYJ prints the results of the program. All other subroutines act in support of the previously noted routines.

PROGRAM USAGE

The program is written in the FORTRAN IV computer language and requires 42000 words of execution field on a CDC CYBER 173 computer. Typical program run times are about 2 seconds. Appendix C gives a sample case. The output listing gives an echo listing of the input data required to generate the results. A user's guide containing the input format, a glossary of all terms used in the input and output, and program restrictions are presented in the computer program and are found at the top of the program listing in Appendix B.

REFERENCES

1. Everhart, J. L.: A Method for Modifying Two-Dimensional Adaptive Wind-Tunnel Walls Including Analytical and Experimental Verification. NASA TP-2081, February 1983.

APPENDIX A

SUMMARY OF PROGRAM ROUTINES AND THEIR FUNCTIONS

ROUTINE NAME	EXTERNAL REFERENCES	FUNCTIONAL STATEMENT
FLEXWAL	AVGDY DELYJ INITIAL INPT NEWYJ RMACHNO SCALE SMOOTH UINFIN VJACK WALLDTA	This is the main or executive routine of a FORTRAN IV computer program for calculating the jack settings of the upper and lower walls of a two-dimensional solid flexible wall wind tunnel.
CURVD	INTRVL SNHCHS	This function subprogram differentiates a curve at a given point using a spline under tension. The subroutine CURV1 should be called earlier to determine certain necessary parameters.
CURVI	INTRVL SNHCHS	This function subprogram integrates a curve specified by a spline under tension between two given limits. The subroutine HMCURV1 should be called earlier to determine certain necessary parameters.
CURV2	INTRVL SNHCHS	This function subprogram interpolates a curve at a given point using a spline under tension. The subroutine HMCURV1 should be called earlier to determine certain necessary parameters.
IBDRY		This function subprogram determines whether a point is on the data surface or a field point.
INTRVL		This function subprogram determines the index of the interval (determined by a given increasing sequence) in which a given value lies.
ISTRIP		This function subprogram determines whether the strip being considered is a normal or singular strip.
RMACHNO		This function subprogram calculates the test Mach numbers.
RMS		This subroutine function determines the RMS value of array A.
SWITCH		This subroutine reorders the indices of array x from high to low.

ROUTINE NAME	EXTERNAL REFERENCES	FUNCTIONAL STATEMENT
UCAL		This function subprogram calculates the U component of velocity.
UINFIN		This function subprogram calculates the free-stream velocity.
AVGDYJ		This subroutine averages the top and bottom wall values of DYJ.
DELYJ	CURVI HMCURV1 RMS	This subroutine integrates the delta theta values to obtain new wall location predictions.
HMCURV1		This subroutine fits a cubic or tension spline through a set of Y vs. X values.
INITIAL		This subroutine presets certain arrays to zero.
INPT		This subroutine handles the input of the raw data as taken from the wind-tunnel test.
LSQSMTH		This subroutine provides a least squares smoothing for either a straight line or a parabolic fit of the input data.
NEWYJ	RMS	This subroutine writes the predicted results onto TAPE 6.
SCALE		This subroutine scales the input using the Goethert scaling rules.
SING		This subroutine evaluates the velocity component along the singular strip.
SMOOTH	LSQSMTH	This subroutine controls the smoothing of the wind-tunnel data.
SNHCSH		This subroutine returns approximations to: SINHM(X) = SINH(X) - X COSHM(X) = COSH(X) - 1 and COSHMM(X) = COSH(X) - 1 - X*X/2 with relative error less than 3.24E-14.
STRIP		This subroutine calculates the jack velocity component resulting from some strip excluding that containing the singularity.

ROUTINE NAME	EXTERNAL REFERENCES	FUNCTIONAL STATEMENT
VJACK	IBDRY ISTRIP STRIP	This subroutine integrates the velocity integral to obtain the jack velocities.
WALLDTA	CURVD CURV2 HMCURV1 SWITCH UCAL	This subroutine takes input data and calculates velocities and wall locations then fills appropriate array locations.

APPENDIX B

PROGRAM LISTING

```
C
C
C*****************************************************************************
C*****************************************************************************
C*                                                                           *
C*    FLEXWAL IS A FORTRAN IV COMPUTER PROGRAM FOR CALCULATING THE JACK      *
C*    SETTINGS OF THE UPPER AND LOWER WALLS OF A TWO-DIMENSIONAL SOLID       *
C*    FLEXIBLE-WALL WIND TUNNEL AS PREDICTED BY THE METHOD OF NASA TP-2081.  *
C*    THE PROGRAM WAS WRITTEN BY JOEL L. EVERHART OF NASA LANGLEY RESEARCH   *
C*    CENTER.                                                                *
C*                                                                           *
C*    REQUIRED INPUT IS READ INTO THE PROGRAM FROM TAPE5 AND OUTPUT IS       *
C*    RETURNED ON TAPE6.                                                     *
C*                                                                           *
C*****************************************************************************
C*                                                                           *
C*    INPUT:                                                                 *
C*                                                                           *
C*    READ ORDER                 DESCRIPTION                    FORMAT       *
C*                                                                           *
C*         1          HEAD(I),I=1,8                            8A10         *
C*         2          TEST,RUN,POINT                           3I5          *
C*         3          RN, ALPHA, TEMP, PT, PINF, MSET          6F10.0       *
C*         4          NJT, NJB, NTW, NDS, NBW, NUS, NRLAX,                  *
C*                    (IS(I),I=1,4), IAVG, IPRT                13I5         *
C*         5          IOPT(I),I=1,4                            4I5          *
C*         6          EC(I),I=1,4                              4F10.0       *
C*         7          RLAX(I),I=1,NRLAX                        8F10.0       *
C*         8          (XJ(I),YJ(I),I=1,NJ)                     2F10.0       *
C*         9          (XD(I),YD(I),UD(I),I=1,ND)               3F10.0       *
C*                                                                           *
C*****************************************************************************
C*                                                                           *
C*    DEFINITION OF TERMS:                                                   *
C*                                                                           *
C*    ALPHA  - MODEL ANGLE OF ATTACK                                         *
C*    DUJ    - PREDICTED CHANGE IN AXIAL JACK PERTURBATION VELOCITY          *
C*    DVJ    - PREDICTED CHANGE IN VERTICAL JACK PERTURBATION VELOCITY       *
C*    DYJ    - PREDICTED CHANGE IN WALL POSITION                             *
C*    EC     - END CONDITIONS ON THE SPLINE THROUGH THE JACK                 *
C*             LOCATIONS.  THE SPLINE IS USED TO DETERMINE WALL POSITION     *
C*             BETWEEN JACKS AND WALL SLOPES.                                *
C*                  EC(1), UPSTREAM,UPPER WALL                               *
C*                  EC(2), DOWNSTREAM, UPPER WALL                            *
C*                  EC(3), UPSTREAM, LOWER WALL                              *
C*                  EC(4), DOWNSTREAM, LOWER WALL                            *
C*    HEAD   - DESCRIPTOR CARD OF UP TO 80 CHARACTERS.                       *
C*    IAVG   - WALL JACK AVERAGING PARAMETER.                                *
C*             AT A GIVEN XJ STATION THE UPPER AND LOWER YJ POSITION ARE     *
```

```
C*              AVERAGED GIVING SYMMETRICAL RESULTS                    *
C*                   IAVG=0, AVERAGING ON                              *
C*                        1, AVERAGING OFF                             *
C*     IOPT   - SPLINE END CONDITION OPTION PARAMETER                   *
C*                   IOPT(1), UPSTREAM, UPPER WALL                     *
C*                   IOPT(2), DOWNSTREAM, UPPER WALL                   *
C*                   IOPT(3), UPSTREAM, LOWER WALL                     *
C*                   IOPT(4), DOWNSTREAM, LOWER WALL                   *
C*                         IOPT(I)=0, EC UNKNOWN                       *
C*                                =1, EC=D(Y)/DX                       *
C*                                =2, EC=D2(Y)/DX2                     *
C*     IPRT   - PRINT CONTROL PARAMETER.                                *
C*                    IPRT=0, MAXIMUM OUTPUT                           *
C*                        =1, NORMAL OUTPUT                            *
C*     IS     - LEAST SQUARES SMOOTHING PARAMETER FOR INPUT DATA ON     *
C*              EACH WALL OF CONTOUR.  IS(I) IS THE NUMBER OF SMOOTHING *
C*              PASSES MADE ON EACH WALL STARTING CLOCKWISE ON UPPER WALL.*
C*              IS(I)=0 IMPLIES NO SMOOTHING                            *
C*     MSET   - DESIRED MACH NUMBER TO WHICH RESULTS ARE SCALED.        *
C*     ND     - TOTAL NUMBER OF PRESSURE MEASUREMENTS (CALCULATED       *
C*              INTERNALLY TO PROGRAM)                                  *
C*     NBW    - NUMBER OF PRESSURE MEASUREMENTS ON BOTTOM WALL (LOWER LEG*
C*              OF CONTOUR)                                             *
C*     NDS    - NUMBER OF PRESSURE MEASUREMENTS ON DOWNSTREAM LEG OF CONTOUR *
C*     NTW    - NUMBER OF PRESSURE MEASUREMENTS ON TOP WALL (UPPER LEG OF*
C*              CONTOUR)                                                *
C*     NUS    - NUMBER OF PRESSURE MEASUREMENTS ON UPSTREAM LEG OF CONTOUR *
C*     NJ     - TOTAL NUMBER OF JACKS (CALCULATED INTERNALLY TO PROGRAM)*
C*     NJB    - NUMBER OF JACKS ON BOTTOM WALL                          *
C*     NJT    - NUMBER OF JACKS ON TOP WALL                             *
C*     NRLAX  - NUMBER OF WALL RELAXATION FACTORS                       *
C*     PINF   - FREESTREAM STATIC PRESSURE, PSI OR PSF                  *
C*     POINT  - DATA IDENTIFIER                                         *
C*     PT     - FREESTREAM TOTAL PRESSURE, PSI OR PSF                   *
C*     RLAX   - VALUE OF WALL CHANGE RELAXATION FACTORS FOR FINAL RESULTS *
C*     RMS    - ROOT-MEAN-SQUARE ERROR                                  *
C*     RN     - TEST REYNOLDS NUMBER                                    *
C*     RUN    - DATA IDENTIFIER                                         *
C*     TEMP   - FREESTREAM TOTAL TEMPERATURE, DEG. RANKINE              *
C*     TEST   - DATA IDENTIFIER                                         *
C*     THETA  - FLOW ANGLE AT WALL                                      *
C*     THETAN - PREDICTED FLOW ANGLE AT WALL                            *
C*     UD     - ON INPUT, MEASURED WALL PRESSURE COEFFICIENTS.  ON OUTPUT,*
C*              CALCULATED AXIAL VELOCITY.                              *
C*     UJ     - AXIAL VELOCITY AT JACK                                  *
C*     UJN    - PREDICTED AXIAL JACK VELOCITY                           *
C*     VJ     - VERTICAL VELOCITY AT JACK                               *
C*     VJN    - PREDICTED VERTICAL VELOCITY AT JACK                     *
C*     XD     - AXIAL STATION OF WALL ORIFICES                          *
C*     XJ     - AXIAL STATION OF WALL JACKS                             *
C*     YD     - NOMINAL VERTICAL LOCATION OF UPPER AND LOWER WALL ORIFICES. *
```

```
C*              ACTUAL VERTICAL LOCATION OF UPSTREAM AND DOWNSTREAM ORIFICES.    *
C*              ACTUAL UPPER AND LOWER WALL ORDINATES ARE CALCULATED FROM A      *
C*              SPLINE THROUGH THE JACK ORDINATES.                               *
C*     YJ       - VERTICAL LOCATION OF WALL JACKS.                               *
C*     YJN      - PREDICTED VERTICAL LOCATION OF WALL JACKS.                     *
C*                                                                               *
C*******************************************************************************
C*                                                                               *
C*     RESTRICTIONS:                                                             *
C*                                                                               *
C*     1. WALL DATA ARE INPUT CLOCKWISE STARTING AT THE MOST UPSTREAM UPPER      *
C*        WALL ORIFICE.                                                          *
C*     2. THE JACK ORDINATES ARE INPUT FROM UPSTREAM TO DOWNSTREAM UPPER WALL    *
C*        AND FROM UPSTREAM TO DOWNSTREAM LOWER WALL.                            *
C*     3. USE CONSISTENT PRESSURE UNITS.                                         *
C*     4. SMOOTHING OF UPSTREAM AND DOWNSTREAM DATA IS NOT INCLUDED,             *
C*        HOWEVER, NOTE IS MADE IN THE LISTING OF SUBROUTINE SMOOTH WHERE        *
C*        IT SHOULD BE INCLUDED IF DESIRED.                                      *
C*     5. THE FLOW ANGLES (IE. VD/UD) ON THE UPSTREAM AND DOWNSTREAM             *
C*        PORTIONS OF THE DATA CONTOURS ARE PRESENTLY SET EQUAL TO THE           *
C*        AVERAGE OF THE SLOPES OF THE UPPER AND LOWER WALLS AT THAT POSITION.   *
C*                                                                               *
C*******************************************************************************
C*******************************************************************************
C
C
C
C
C
C
      PROGRAM FLEXWAL(INPUT,OUTPUT,TAPE5,TAPE6)
      REAL M,MSET
      INTEGER TEST,RUN,POINT
      COMMON/JACK/NJ,NJT,NJR,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/PARAM/HEAD(8),TEST,RUN,POINT,M,RN,ALPHA,TEMP,RLAX(10),
     $NRLAX,PT,PINF,UINF,IAVG,IS(4),IOPT(4),EC(4),GAMMA,MSET
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      DIMENSION WK1(1000),WK2(100)
      GAMMA=1.4
      GASR=1716.0
      NWK1=1000
      NWK2=100
      CALL INITIAL(NWK2,WK2)
      CALL INPT(IPRT)
      M=RMACHNO(GAMMA,PT,PINF)
      UINF=UINFIN(GAMMA,GASR,TEMP,M)
      WRITE(6,96)M,UINF
      IF(IPRT.EQ.0)WRITE(6,99)
```

```
      CALL SCALE(NJ,XJ,YJ,DYJ,UJ,VJ,M,MSET,0,1,IPRT)
      IF(IPRT.EQ.0)WRITE(6,98)
      CALL SCALE(ND,XD,YD,WK2,UD,VD,M,MSET,0,2,IPRT)
      CALL SMOOTH(500,WK1(1),WK1(501),IS,IPRT)
      CALL WALLDTA(NWK1,WK1,IPRT)
      CALL VJACK(IOPT)
      CALL DELYJ(NWK2,WK2,NWK1,WK1,UINF)
      IF(IPRT.EQ.0)WRITE(6,97)
      CALL SCALE(NJ,XJ,YJ,DYJ,UJN,VJN,M,MSET,1,3,IPRT)
      IF(IAVG.EQ.0)CALL AVGDYJ(NJ,XJ,DYJ)
      DO 25 I=1,NRLAX
      CALL NEWYJ(I,NWK2,WK2)
   25 CONTINUE
      STOP
   96 FORMAT(///* M,INF=*G12.6*   UINF=*G12.6//)
   97 FORMAT("1"//*     CALCULATED JACK DATA*)
   98 FORMAT("1"//*     INPUT WALL DATA*)
   99 FORMAT("1"//*     INPUT JACK DATA*)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION CALCULATES THE TEST MACH NUMBER
C
      FUNCTION RMACHNO(GAMMA,PT,PINF)
      REAL M2
      PR=(PT/PINF)**((GAMMA-1.0)/GAMMA)
      M2=(PR-1.0)*2.0/(GAMMA-1.0)
      IF(M2.LT.0.0)M2=0.0
      RMACHNO=SQRT(M2)
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION CALCULATES THE FREESTREAM VELOCITY
C
      FUNCTION UINFIN(GAMMA,GASR,TEMP,M)
      REAL M
      TINF=TEMP/(1.0+(GAMMA-1.0)*M*M/2.0)
      A2=GAMMA*GASR*TINF
      A=SQRT(A2)
      UINFIN=A*M
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE PRESETS CERTAIN ARRAYS TO ZERO
C
      SUBROUTINE INITIAL(NWK2,WK2)
```

```
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      DIMENSION WK2(NWK2)
      DO 10 I=1,100
      XJ(I)=0.0
      YJ(I)=0.0
      UJ(I)=0.0
      VJ(I)=0.0
      UJN(I)=0.0
      VJN(I)=0.0
      YJN(I)=0.0
      DYJ(I)=0.0
      XD(I)=0.0
      YD(I)=0.0
      UD(I)=0.0
      VD(I)=0.0
      WK2(I)=0.0
   10 CONTINUE
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C     THIS SUBROUTINE HANDLES THE INPUT OF THE RAW DATA AS TAKEN FROM
C     THE WIND TUNNEL TEST
C
      SUBROUTINE INPT(IPRT)
      REAL M,MSET
      INTEGER TEST,RUN,POINT
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJO(100),VJO(100),
     $UJ(100),VJ(100),YJN(100)
      COMMON/PARAM/HEAD(8),TEST,RUN,POINT,M,RN,ALPHA,TEMP,RLAX(10),
     $NRLAX,PT,PINF,UINF,IAVG,IS(4),IOPT(4),EC(4),GAMMA,MSET
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      REWIND 5
      READ(5,290)(HEAD(I),I=1,8)
      READ(5,291)TEST,RUN,POINT
      READ(5,292)RN,ALPHA,TEMP,PT,PINF,MSET
      READ(5,291) NJT,NJB,NTW,NDS,NBW,NUS,NRLAX,(IS(I),I=1,4),IAVG,IPRT
      NJ=NJT+NJB
      ND=NTW+NDS+NBW+NUS
      READ(5,291)(IOPT(I),I=1,4)
      READ(5,292)(EC(I),I=1,4)
      READ(5,292)(RLAX(I),I=1,NRLAX)
      READ(5,294)(XJ(I),YJ(I),I=1,NJ)
      READ(5,299)(XD(I),YD(I),UD(I),I=1,ND)
C
```

```
C*****ECHO CHECK OF INPUT DATA
C
      WRITE(6,295)
      WRITE(6,296)(HEAD(I),I=1,8)
      WRITE(6,297)TEST,RUN,POINT
      WRITE(6,298)RN,ALPHA,TEMP,PT,PINF,MSET
      WRITE(6,297)NJT,NJB,NTW,NDS,NBW,NUS,NRLAX,(IS(I),I=1,4),IAVG,IPRT
      WRITE(6,297)(IOPT(I),I=1,4)
      WRITE(6,298)(EC(I),I=1,4)
      WRITE(6,298)(RLAX(I),I=1,NRLAX)
      WRITE(6,288)(XJ(I),YJ(I),I=1,NJ)
      WRITE(6,287)(XD(I),YD(I),UD(I),I=1,ND)
      RETURN
  287 FORMAT(3X,3F10.5)
  288 FORMAT(3X,2F10.5)
  290 FORMAT(8A10)
  291 FORMAT(16I5)
  292 FORMAT(8F10.0)
  294 FORMAT(2F10.0)
  295 FORMAT(1H1,//,*       INPUT DATA LISTING       *,/)
  296 FORMAT(3X,8A10)
  297 FORMAT(3X,16I5)
  298 FORMAT(3X,8G11.5)
  299 FORMAT(3F10.0)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE CONTROLS SMOOTHING OF THE WIND TUNNEL DATA
C
      SUBROUTINE SMOOTH(NWK,WK1,WK2,IS,IPRT)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      DIMENSION IS(4),WK1(NWK),WK2(NWK),DUM(10)
      N1=IS(1)
      N2=IS(2)
      N3=IS(3)
      N4=IS(4)
C*****SMOOTH TOP WALL DATA
      IF(N1.EQ.0)GO TO 26
      DO 5 I=1,NTW
      WK1(I)=XD(I)
      WK2(I)=UD(I)
    5 CONTINUE
      DO 15 I=1,N1
      CALL LSQSMTH(5,WK1(1),WK2(1),5,DUM,1,WK1(1),WKS,0)
      WK2(1)=WKS
      CALL LSQSMTH(5,WK1(1),WK2(1),5,DUM,1,WK1(2),WKS,0)
      WK2(2)=WKS
      NS=NTW-2
      DO 10 J=3,NS
      IJ=J-2
```

```
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(J),WKS,0)
      WK2(J)=WKS
   10 CONTINUE
      IJ=NTW-4
      NS=NS+1
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(NS),WKS,0)
      WK2(NS)=WKS
      NS=NS+1
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(NS),WKS,0)
      WK2(NS)=WKS
   15 CONTINUE
      IF(IPRT.EQ.1)GO TO 26
      DO 20 I=1,NTW
      WRITE(6,100)XD(I),UD(I),WK2(I)
      XD(I)=WK1(I)
      UD(I)=WK2(I)
   20 CONTINUE
C*****SMOOTH DOWNSTREAM DATA
   26 IF(N2.EQ.0)GO TO 51
C*****SMOOTH BOTTOM WALL DATA
   51 IF(N3.EQ.0)GO TO 76
      DO 55 I=1,NBW
      II=NTW+NDS+NBW+1-I
      WK1(I)=XD(II)
      WK2(I)=UD(II)
   55 CONTINUE
      DO 65 I=1,N3
      CALL LSQSMTH(5,WK1(1),WK2(1),5,DUM,1,WK1(1),WKS,0)
      WK2(1)=WKS
      CALL LSQSMTH(5,WK1(1),WK2(1),5,DUM,1,WK1(2),WKS,0)
      WK2(2)=WKS
      NS=NBW-2
      DO 50 J=3,NS
      IJ=J-2
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(J),WKS,0)
      WK2(J)=WKS
   50 CONTINUE
      IJ=NBW-4
      NS=NS+1
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(NS),WKS,0)
      WK2(NS)=WKS
      NS=NS+1
      CALL LSQSMTH(5,WK1(IJ),WK2(IJ),5,DUM,1,WK1(NS),WKS,0)
      WK2(NS)=WKS
   65 CONTINUE
      IF(IPRT.EQ.1)GO TO 76
      DO 70 I=1,NBW
      II=NTW+NDS+NBW+1-I
      WRITE(6,100)XD(II),UD(II),WK2(I)
```

```
      XD(II)=WK1(I)
      UD(II)=WK2(I)
   70 CONTINUE
C*****SMOOTH UPSTREAM DATA
   76 IF(N4.EQ.0)GO TO 99
   99 RETURN
  100 FORMAT(*  XD=*F10.5*  UD=*F10.5*  UD(SMOOTHED)=*F10.5)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE LSQSMTH(M,X,Y,NC,C,MS,XS,YS,IC)
C
C*******************************************************************************
C
C*    LSQSMTH IS A LEAST SQUARES SMOOTHING ROUTINE FOR EITHER A STRAIGHT       *
C*    LINE OR A PARABOLIC FIT OF THE INPUT DATA                                *
C*                                                                             *
C*    M------NUMBER OF DATA POINTS USED IN THE SMOOTHING                       *
C*    X,Y----1-D ARRAYS OF COORDINATES OF DATA TO BE SMOOTHED                  *
C*    NC-----NUMBER OF LEAST SQUARES COEFFICIENTS TO BE COMPUTED PLUS 2        *
C*           NC=4---STRAIGHT LINE FIT                                          *
C*           NC=5---PARABOLIC FIT                                              *
C*    C------1-D ARRAY OF LENGTH N+2 FOR COEFFICIENTS                          *
C*    MS-----NUMBER OF POINTS TO BE INTERPOLATED AFTER FITTING DATA            *
C*    XS-----1-D ARRAY OF LENGTH MS CONTAINING LOCATION OF POINT TO BE         *
C*           SMOOTHED                                                          *
C*    YS-----1-D ARRAY OF LENGTH MS CONTAINING SMOOTHED ORDINATE OF XS         *
C*    IC-----IC=0---COMPUTE COEFFICIENTS                                       *
C*           IC=1---COEFFICIENTS PREVIOUSLY COMPUTED AND SUPPLIED              *
C*                                                                             *
C*******************************************************************************
C
      DIMENSION X(M),Y(M),C(NC),XS(MS),YS(MS)
      N=NC-2
      IF(M.LT.N)GO TO 50
      IF(IC.EQ.1)GO TO 21
      SY=SX=SXY=SX2=SX2Y=SX3=SX4=0.0
      DO 3 I=1,M
      SX=SX+X(I)
      SY=SY+Y(I)
    3 CONTINUE
      C(N+1)=SX/M  $  C(N+2)=SY/M
      DO 5 I=1,M
      X(I)=X(I)-C(N+1)
      Y(I)=Y(I)-C(N+2)
    5 CONTINUE
      DO 10 I=1,M
      SXY=SXY+X(I)*Y(I)
      SX2=SX2+X(I)*X(I)
      IF(N.EQ.2)GO TO 10
```

```
      SX2Y=SX2Y+X(I)*X(I)*Y(I)
      SX3=SX3+X(I)*X(I)*X(I)
      SX4=SX4+X(I)*X(I)*X(I)*X(I)
   10 CONTINUF
      IF(N.EQ.2)GO TO 15
      C(3)=(SX2Y*SX2-SXY*SX3)/(SX2*SX4-SX2*SX2*SX2/M-SX3*SX3)
      C(2)=(SXY-C(3)*SX3)/SX2
      C(1)=-C(3)*SX2/M
      GO TO 18
   15 C(2)=SXY/SX2
      C(1)=0.0
   18 DO 20 I=1,M
      X(I)=X(I)+C(N+1)
      Y(I)=Y(I)+C(N+2)
   20 CONTINUF
   21 IF(MS.EQ.0)GO TO 35
      DO 23 I=1,MS
      XS(I)=XS(I)-C(N+1)
   23 CONTINUF
      DO 30 I=1,MS
      YS(I)=0.0
      DO 25 J=1,N
      YS(I)=YS(I)*XS(I)+C(N+1-J)
   25 CONTINUF
   30 CONTINUF
      DO 33 I=1,MS
      XS(I)=XS(I)+C(N+1)
      YS(I)=YS(I)+C(N+2)
   33 CONTINUF
   35 RETURN
   50 PRINT 51,M,N
   51 FORMAT("1",/////,*   M.LT.N IN LSQSMTH.  M=*,I3,*   N=*I3)
      STOP
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SURROUTINE TAKES INPUT DATA AND CALCULATES VELOCITIFS AND
C     WALL LOCATIONS THEN FILLS APPPROPRIATE ARRAY LOCATIONS.
C
      SUBROUTINE WALLDTA(NYPP,YPP,IPRT)
      REAL M,MSET
      INTEGER TEST,RUN,POINT
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/PARAM/HEAD(8),TEST,RUN,POINT,M,RN,ALPHA,TEMP,RLAX(10),
     $NRLAX,PT,PINF,UINF,IAVG,IS(4),IOPT(4),EC(4),GAMMA,MSET
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      DIMENSION YPP(NYPP),JOPT(2)
```

```
C
C*****SPLINE FIT OF UPPER WALL JACK COORDINATES FOR YD, (VD/UD), AND (VJ/UJ)
C
      CALL HMCURV1(XJ(1),YJ(1),YPP,NJT,0.0,IOPT(1),EC(1),FC(2),IERR)
      DO 305 I=1,NTW
      YD(I)=CURV2(XD(I),NJT,XJ(1),YJ(1),YPP,0.0)
      VD(I)=CURVD(XD(I),NJT,XJ(1),YJ(1),YPP,0.0)
  305 CONTINUE
      DO 306 I=1,NJT
      VJ(I)=CURVD(XJ(I),NJT,XJ(1),YJ(1),YPP,0.0)
  306 CONTINUE
C
C*****SPLINE FIT OF LOWER WALL JACK COORDINATES FOR YD, (VD/UD), AND (VJ/UJ)
C
      CALL HMCURV1(XJ(NJT+1),YJ(NJT+1),YPP,NJB,0.0,IOPT(3),EC(3),FC(4),
     *IERR)
      DO 307 I=1,NBW
      II=NTW+NDS+I
      YD(II)=CURV2(XD(II),NJB,XJ(NJT+1),YJ(NJT+1),YPP,0.0)
      VD(II)=CURVD(XD(II),NJB,XJ(NJT+1),YJ(NJT+1),YPP,0.0)
  307 CONTINUE
      DO 308 I=1,NJB
      II=NJT+I
      VJ(II)=CURVD(XJ(II),NJB,XJ(NJT+1),YJ(NJT+1),YPP,0.0)
  308 CONTINUE
C
C*****CALCULATION OF (VD/UD) COMPONENT ALONG THE DOWNSTREAM END
C
      VDAVG=(VD(NTW)+VD(NTW+NDS+1))/2.0
      DO 315 I=1,NDS
      II=NTW+I
      VD(II)=VDAVG
  315 CONTINUE
C
C*****CALCULATION OF (VD/UD) COMPONENT ALONG THE UPSTREAM END
C
      VDAVG=(VD(NTW+NDS+NBW)+VD(1))/2.0
      DO 320 I=1,NUS
      II=NTW+NDS+NBW+I
      VD(II)=VDAVG
  320 CONTINUE
C
C*****CALCULATION OF UD AND VD VELOCITIES
C
      DO 325 I=1,ND
      UD(I)=UCAL(1.,UINF,UD(I),VD(I),GAMMA)
      VD(I)=VD(I)*UD(I)
  325 CONTINUE
C
```

```
C*****CALCULATION OF UJ AND VJ FROM UD DATA
C
      JOPT(1)=1
      JOPT(2)=1
      EC1=0.0
      ECN=0.0
      ST=0.0
      CALL HMCURV1(XD(1),UD(1),YPP,NTW,ST,JOPT,EC1,ECN,IERR)
      DO 340 I=1,NJT
      UJ(I)=CURV2(XJ(I),NTW,XD(1),UD(1),YPP,ST)
      VJ(I)=VJ(I)*UJ(I)
  340 CONTINUE
      IB=NTW+NDS+1
      CALL SWITCH(NBW,XD(IB))
      CALL SWITCH(NBW,UD(IB))
      CALL SWITCH(NBW,VD(IB))
      JOPT(1)=1
      JOPT(2)=1
      EC1=0.0
      ECN=0.0
      CALL HMCURV1(XD(IB),UD(IB),YPP,NBW,ST,JOPT,EC1,ECN,IERR)
      DO 350 I=1,NJB
      UJ(NJT+I)=CURV2(XJ(NJT+I),NBW,XD(IB),UD(IB),YPP,ST)
      VJ(NJT+I)=VJ(NJT+I)*UJ(NJT+I)
  350 CONTINUE
      CALL SWITCH(NBW,XD(IB))
      CALL SWITCH(NBW,UD(IB))
      CALL SWITCH(NBW,VD(IB))
C
C*****REMOVE FREESTREAM COMPONENT FROM UD AND UJ TO OBTAIN
C*****THE PERTURBATION QUANTITIES
C
      DO 355 I=1,NJ
      UJ(I)=UJ(I)-UINF
  355 CONTINUE
      DO 360 I=1,ND
      UD(I)=UD(I)-UINF
  360 CONTINUE
C
C*****CLOSE DATA CONTOUR
C
      ND=ND+1
      XD(ND)=XD(1)
      YD(ND)=YD(1)
      UD(ND)=UD(1)
      VD(ND)=VD(1)
C
C*****LOOK AT RESULTS OF THIS SUBROUTINE
C
```

```
      IF(IPRT.EQ.1)GO TO 390
      WRITE(6,398)UINF
      DO 365 I=1,NJ
      WRITE(6,399)I,XJ(I),YJ(I),UJ(I),VJ(I)
  365 CONTINUE
      WRITE(6,397)
      DO 370 I=1,ND
      WRITE(6,399)I,XD(I),YD(I),UD(I),VD(I)
  370 CONTINUE
  390 RETURN
  397 FORMAT("1"//*        WALL DATA (SCALED)*//5X*I*
     $18X*XD*18X*YD*18X*UD*18X*VD*/)
  398 FORMAT("1"//7X*UINF=*G12.6//7X*JACK DATA (SCALED)*//5X*I*
     $18X*XJ*18X*YJ*18X*UJ*18X*VJ*/)
  399 FORMAT(3XI3,4(5XF15.9))
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION CALCULATES THE U COMPONENT OF VELOCITY
C
      FUNCTION UCAL(M,UINF,CP,THETA,GAMMA)
      REAL M
      T2=1.0+THETA*THETA
      IF(M.LE.0.0)GO TO 10
      P=(GAMMA-1.0)/GAMMA
      A1=1.0-(1.0+GAMMA*M*M*CP/2.0)**P
      A2=1.0+2.0*A1/((GAMMA-1.0)*M*M)
      UCAL=UINF*SQRT(A2/T2)
      RETURN
   10 A2=1.0-CP
      IF(A2.LT.0.0)A2=0.0
      UCAL=UINF*SQRT(A2/T2)
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION REORDERS THE INDICES OF ARRAY X FROM HIGH TO LOW
C
      SUBROUTINE SWITCH(N,X)
      DIMENSION X(N)
      DO 10 I=1,N
      J=N+1-I
      IF(J.LE.I)GO TO 11
      S=X(J)
      X(J)=X(I)
      X(I)=S
   10 CONTINUE
   11 RETURN
      END
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE SCALES THE INPUT USING THE GOETHERT SCALING RULES
C
      SUBROUTINE SCALE(N,X,Y,DY,U,V,M,MSET,IS,IT,IPRT)
      REAL M,MSET
      DIMENSION X(N),Y(N),U(N),V(N),DY(N)
      BETA2=1.0-M*M
      BETA=SQRT(BETA2)
      BETAMS=SQRT(1.0-MSET*MSET)
C
C*****IS=0---COMPRESSIBLE TO INCOMPRESSIBLE
C*****IS=1---INCOMPRESSIBLE TO COMPRESSIBLE
C
      IF(IS.EQ.0.AND.IPRT.EQ.0)WRITE(6,25)
      IF(IS.EQ.1.AND.IPRT.EQ.0)WRITE(6,26)
      IF(IT.EQ.1.AND.IPRT.EQ.0)WRITE(6,28)
      IF(IT.EQ.2.AND.IPRT.EQ.0)WRITE(6,29)
      IF(IT.EQ.3.AND.IPRT.EQ.0)WRITE(6,30)
      IF(IS.EQ.0)GO TO 5
      BETAMS=1.0/BETAMS
      BETA=1.0/BETA
      BETA2=1.0/BETA2
    5 DO 10 I=1,N
      X(I)=X(I)
      Y(I)=BETA*Y(I)
      IF(IS.EQ.1)DY(I)=BETAMS*DY(I)
      U(I)=BETA2*U(I)
      V(I)=BETA*V(I)
      IF(IT.EQ.1.AND.IPRT.EQ.0)WRITE(6,27)I,X(I),Y(I)
      IF(IT.EQ.2.AND.IPRT.EQ.0)WRITE(6,27)I,X(I),Y(I),U(I)
      IF(IT.EQ.3.AND.IPRT.EQ.0)WRITE(6,27)I,X(I),Y(I),DY(I),U(I),V(I)
   10 CONTINUE
      RETURN
   25 FORMAT(//*     COMPRESSIBLE TO INCOMPRESSIBLE SCALING*)
   26 FORMAT(//*     INCOMPRESSIBLE TO COMPRESSIBLE SCALING*)
   27 FORMAT(2XI3,5(2X,G12.6))
   28 FORMAT(//4X*I*12X*XJ*12X*YJ*/)
   29 FORMAT(//4X*I*12X*XD*12X*YD*12X*CP*/)
   30 FORMAT(//4X*I*12X*XJ*12X*YJ*12X*DY*12X*UJ*12X*VJ*/)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE INTEGRATES THE VELOCITY INTEGRAL TO OBTAIN THE JACK VELOCITY
C
      SUBROUTINE VJACK(IOPT)
      DIMENSION IOPT(4)
```

```
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NPW,NUS,XD(100),YD(100),UD(100),VD(100)
      PI=ATAN2(0.0,-1.0)
      IVJ=0
      NDM1=ND-1
      DO 40 J=1,NJ
      IF(J.EQ.IVJ)WRITE(6,96)
      VJN(J)=0.0
      UJN(J)=0.0
      DO 30 I=1,NDM1
      ISTR=ISTRIP(J,I)
      IF(ISTR.EQ.1)CALL STRIP(J,I,I+1,USTRIP,VSTRIP)
      IF(ISTR.EQ.2)CALL SING(J,I,USTRIP,VSTRIP)
      VJN(J)=VJN(J)+VSTRIP
      UJN(J)=UJN(J)-USTRIP
      IF(J.EQ.IVJ)WRITE(6,99)J,I,XJ(J),YJ(J),XD(I),YD(I),
     $XD(I+1),YD(I+1),VSTRIP,VJN(J),ISTR,USTRIP,UJN(J)
30    CONTINUE
      PIDIV=PI
      IF(IBDRY(J).EQ.2)PIDIV=2.0*PI
      VJN(J)=VJN(J)/PIDIV
      UJN(J)=UJN(J)/PIDIV
40    CONTINUE
      IF(IOPT(1).EQ.1)GO TO 42
      VJN(1)=VJ(1)
      UJN(1)=UJ(1)
42    IF(IOPT(2).EQ.1)GO TO 44
      VJN(NJT)=VJ(NJT)
      UJN(NJT)=UJ(NJT)
44    IF(IOPT(3).EQ.1)GO TO 46
      VJN(NJT+1)=VJ(NJT+1)
      UJN(NJT+1)=UJ(NJT+1)
46    IF(IOPT(4).EQ.1)GO TO 48
      VJN(NJT+NJB)=VJ(NJT+NJB)
      UJN(NJT+NJB)=UJ(NJT+NJB)
48    WRITE(6,97)
      DO 50 I=1,NJ
      DVJ=VJN(I)-VJ(I)
      DUJ=UJN(I)-UJ(I)
      WRITE(6,98)I,XJ(I),YJ(I),UJ(I),UJN(I),VJ(I),VJN(I),DVJ,DUJ
50    CONTINUE
      RETURN
96    FORMAT("1")
97    FORMAT("1"//*     NEW JACK PERTURBATION VELOCITY CALCULATIONS*
     $* (SCALED)*//*  I*
     $6X*XJ*8X*YJ*9X*UJ*6X*UJN*5X*VJ*9X*VJN*9X*DVJ*9X*DUJ*/)
98    FORMAT(I3,2(2XF8.4),2(2XF7.2),4(2XG10.4))
99    FORMAT(* J=*I2* I=*I2* XJ=*F7.3* YJ=*F8.4* XD=*F7.3* YD=*F8.4
     $* XD=*F7.3* YD=*F8.4* VSTRIP=*G12.6* VJN=*G12.6* ISTRIP=*I2/
     $79X* USTRIP=*G12.6* UJN=*G12.6)
```

```
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE CALCULATES THE JACK VELOCITY COMPONENT RESULTING FROM
C     SOME STRIP EXCLUDING THAT CONTAINING THE SINGULARITY
C
      SUBROUTINE STRIP(I,K,L,USTRIP,VSTRIP)
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
      XL=XD(L)-XD(K)  $  YL=YD(L)-YD(K)  $  SL=SQRT(XL*XL+YL*YL)
      RX=XJ(I)-XD(K)  $  RY=YJ(I)-YD(K)
      UDIF=UD(L)-UD(K)  $  VDIF=VD(L)-VD(K)
      A1=-RX*UD(K)  $  A2=-RX*VD(K)  $  A3=-RY*UD(K)  $  A4=-RY*VD(K)
      B1=(-RX*UDIF+XL*UD(K))/SL  $  B2=(-RX*VDIF+XL*VD(K))/SL
      B3=(-RY*UDIF+YL*UD(K))/SL  $  B4=(-RY*VDIF+YL*VD(K))/SL
      SL2=SL*SL
      C1=XL*UDIF/SL2  $  C2=XL*VDIF/SL2
      C3=YL*UDIF/SL2  $  C4=YL*VDIF/SL2
      D1=RX*RX+RY*RY  $  D2=-2*(RX*XL+RY*YL)/SL
      E1=XL/SL  $  E2=YL/SL
      AD=A1-A4  $  BD=B1-B4  $  CD=C1-C4
      AP=A3+A2  $  BP=B3+B2  $  CP=C3+C2
      Q=4.*D1-D2*D2
      IF(Q)630,610,605
  605 QRT=SQRT(Q)
      T1=2.*ATAN(SL*QRT/(2.0*D1+SL*D2))/QRT
      ARG=(D1+D2*SL+SL2)/D1
      IF(ARG.LE.0.0)GO TO 650
      T2=.5*ALOG(ARG)-.5*D2*T1
      T3=SL-D2*T2-D1*T1
      GO TO 620
  610 T1=2.0*SL/(D2*(D2/2.0+SL))
      ARG=2.*(D2/2.0+SL)/D2
      IF(ARG.LE.0.0)GO TO 650
      T2=ALOG(ARG)-.5*D2*T1
      T3=SL-D2*T2-(D2/2.)*(D2/2.)*T1
  620 VSTRIP=(E1*(AD*T1+BD*T2+CD*T3)+E2*(AP*T1+BP*T2+CP*T3))
      USTRIP=(E1*(AP*T1+BP*T2+CP*T3)-E2*(AD*T1+BD*T2+CD*T3))
      RETURN
  630 PRINT 640,Q
  640 FORMAT(//,*ERROR---THE VALUE OF Q IS NEGATIVE, Q=*,F15.8,//)
      STOP
  650 PRINT 651,I,K,L,ARG
  651 FORMAT(/,* I=*,I2,* J=*,I2,* J+1=*,I2,* ARG=*,F25.15,
     $/,* ERROR---ARGUMENT OF LOG IN STRIP.LT.0*)
      STOP
      END
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SURROUTINE EVALUATES THE VELOCITY COMPONENT ALONG THE SINGULAR STRIP
C
      SUBROUTINE SING(I,J,USING,VSING)
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
C     PC=SQRT((XJ(I)-XD(J))**2+(YJ(I)-YD(J))**2)
      XJX1=XJ(I)-XD(J)
      YJY1=XJX1*(YD(J+1)-YD(J))/(XD(J+1)-XD(J))
      PC=SQRT(YJY1*YJY1+XJX1*XJX1)
      SL=SQRT((XD(J+1)-XD(J))**2+(YD(J+1)-YD(J))**2)
      ARG=SL/PC-1.0
      IF(ARG.LE.0.)GO TO 750
      T1=UD(J+1)-UD(J)
      T2=UD(J)+T1*PC/SL
      T3=ALOG(ARG)
      T4=VD(J+1)-VD(J)
      T5=VD(J)+T4*PC/SL
      VSING=T1+T2*T3
      USING=T4+T5*T3
      RETURN
  750 PRINT 751,I,J,ARG
  751 FORMAT(* I=*,I2,* J=*,I2,* ARG=*,F25.15,/,* ERROR---
     $ARGUMENT OF LOG IN SING LE 0*)
      STOP
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION DETERMINES WHETHER THE STRIP BEING CONSIDERED IS A
C     NORMAL OR SINGULAR STRIP
C
      FUNCTION ISTRIP(IJ,ID)
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
C
C*****ISTRIP=1---NONSINGULAR STRIP
C*****ISTRIP=2---SINGULAR STRIP
C
      ISTRIP=1
      SJ=XJ(IJ)
      TJ=YJ(IJ)
      S1=XD(ID)
      T1=YD(ID)
      S2=XD(ID+1)
      T2=YD(ID+1)
      DIF=S2-S1
```

```
      PROD=T1*TJ*T2
      IF(DIF)10,30,20
   10 IF((SJ.GT.S2.AND.SJ.LT.S1).AND.(PROD.LT.0.0))ISTRIP=2
      RETURN
   20 IF((SJ.GT.S1.AND.SJ.LT.S2).AND.(PROD.GT.0.0))ISTRIP=2
   30 CONTINUE
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION DETERMINES WHETHER A POINT IS A POINT ON THE DATA SURFACE
C     OR A FIELD POINT
C
      FUNCTION IBDRY(IJ)
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/CONTOUR/ND,NTW,NDS,NBW,NUS,XD(100),YD(100),UD(100),VD(100)
C
C*****IBDRY=1---(XJ,YJ) ON DATA SURFACE
C*****IBDRY=2---(XJ,YJ) IS FIELD POINT
C
      IBDRY=1
      SJ=XJ(IJ)
      IF((SJ.LT.XD(1)).OR.(SJ.GT.XD(NTW)))IBDRY=2
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE INTEGRATES THE DELTA THETA VALUES TO OBTAIN NEW
C     WALL LOCATION PREDICTIONS
C
      SUBROUTINE DELYJ(NTHETA,THETA,NYPP,YPP,UINF)
      DIMENSION THETA(NTHETA),YPP(NYPP),JOPT(2)
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      JOPT(1)=1
      JOPT(2)=1
      EC1=0.0
      ECN=0.0
      DO 10 I=1,NJ
      UJN(I)=UJN(I)+UINF
      UJ(I)=UJ(I)+UINF
      THETA(I)=VJN(I)/UJN(I)
      YJN(I)=VJ(I)/UJ(I)
   10 CONTINUE
      CALL HMCURV1(XJ,THETA,YPP,NJT,0.0,JOPT,EC1,ECN,IERR)
      DO 15 I=1,NJT
```

```
      DYJ(I)=CURVI(XJ(1),XJ(I),NJT,XJ,THETA,YPP,0.0)
   15 CONTINUE
      CALL HMCURV1(XJ,YJN,YPP,NJT,0.0,JOPT,EC1,ECN,IERR)
      DO 20 I=1,NJT
      DYJ(I)=DYJ(I)-CURVI(XJ(1),XJ(I),NJT,XJ,YJN,YPP,0.0)
   20 CONTINUE
      N1=NJT+1
      N2=NJT+NJB
      CALL HMCURV1(XJ(N1),THETA(N1),YPP,NJB,0.0,JOPT,EC1,ECN,IERR)
      DO 25 I=N1,N2
      DYJ(I)=CURVI(XJ(N1),XJ(I),NJB,XJ(N1),THETA(N1),YPP,0.0)
   25 CONTINUE
      CALL HMCURV1(XJ(N1),YJN(N1),YPP,NJB,0.0,JOPT,EC1,ECN,IERR)
      DO 30 I=N1,N2
      DYJ(I)=DYJ(I)-CURVI(XJ(N1),XJ(I),NJB,XJ(N1),YJN(N1),YPP,0.0)
   30 CONTINUE
      WRITE(6,98)
      DO 40 I=1,NJ
      WRITE(6,99)I,XJ(I),YJ(I),UJ(I),UJN(I),VJ(I),VJN(I),YJN(I),THETA(I)
     $,DYJ(I)
   40 CONTINUE
      RMSDYT=RMS(NJT,DYJ)
      RMSDYB=RMS(NJB,DYJ(NJT+1))
      WRITE(6,97) RMSDYT,RMSDYB
      RETURN
   97 FORMAT(//*  RMS(DYJ(TOP))=  *G12.6*   RMS(DYJ(BOT))=  *G12.6)
   98 FORMAT("1"//*      PREDICTED WALL CORRECTION (SCALED)*//
     $2X*I*6X*XJ*8X*YJ*9X*UJ*7X*UJN*5X*VJ*9X*VJN*9X*THETA*6X
     $*THETAN*8X*DYJ*/)
   99 FORMAT(I3,2(2XF8.4),2(2XF7.2),5(2XG10.4))
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS FUNCTION STATEMENTS DETERMINES THE RMS VALUE OF ARRAY A
C
      FUNCTION RMS(N,A)
      DIMENSION A(N)
      RMS=0.0
      DO 10 I=1,N
      RMS=RMS+A(I)*A(I)
   10 CONTINUE
      RMS=SQRT(RMS/N)
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE HMCURV1(X,Y,YPP,N,ST,OPT,EC1,ECN,IERR)
C
C        THIS SUBROUTINE FITS A CUBIC OR TENSION SPLINE THROUGH A SET
```

```
C         OF Y VS X VALUES.
C
C           AUTHOR -- HARRY MORGAN  NASA/LANGLEY  LSAB/STAD
C                     (MODIFIED BY JOEL EVERHART  NASA/LANGLEY)
C
C*************************************************************************
C*                                                                       *
C*      DESCRIPTION OF INPUT AND OUTPUT FOR SUBROUTINE SPLINE            *
C*                                                                       *
C*  PARAMETER                          DESCRIPTION                       *
C*                                                                       *
C*       X  -  INPUT ARRAY OF INDEPENDENT VARIABLE.                      *
C*       Y  -  INPUT ARRAY OF DEPENDENT VARIABLE.                        *
C*     YPP  -  DUMMY WORK ARRAY OF DIMENSION 10*N ON INPUT.              *
C*             ON OUTPUT, ARRAY OF SECOND DERIVATIVES.                   *
C*       N  -  NUMBER OF INPUT VALUES OF X AND Y.                        *
C*      ST  -  SPLINE TENSION (ST=0.0 SAME AS CUBIC SPLINE).             *
C*     OPT  -  END CONDITION OPTION ARRAY OF LENGTH 2.  OPT(1)           *
C*             CORRESPONDS TO CONDITION AT X(1), OPT(2) CORRESPONDS      *
C*             TO CONDITION AT X(N).                                     *
C*             1 - EC1 OR  ECN UNKNOWN                                   *
C*             2 - EC1=YP(1)  OR   ECN=YP(N)                             *
C*             3 - EC1=YPP(1) OR   ECN=YPP(N)                            *
C*     ECN  -  END CONDITION AT END OF Y VS X CURVE.                     *
C*    IERR  -  ERROR CODE                                                *
C*             0 - NO ERROR                                              *
C*              1 - N IS LESS THAN 3                                     *
C*             2 - X IS NOT MONOTONICALLY INCREASING                     *
C*                                                                       *
C*  RESTRICTIONS                                                         *
C*       X MUST BE MONOTONICALLY INCREASING                              *
C*       N MUST BE GREATER THAN OR EQUAL TO 3.                           *
C*        X AND Y         MUST BE DIMENSIONED BY AT LEAST N IN CALLING   *
C*                        PROGRAM.                                       *
C*       ST SHOULD BE NO GREATER THAN 50.                                *
C*                                                                       *
C*************************************************************************
C
      DIMENSION X(N), Y(N), YPP(1), OPT(2)
C
      INTEGER OPT
C
      SINH(X)=0.5*(EXP(X)-EXP(-X))
      COSH(X)=0.5*(EXP(X)+EXP(-X))
C
C        CHECK ON MONOTONICALLY INCREASING X
C
      IERR=0
      IF (N.LT.3) GO TO 32
```

```
      DO 1 I=2,N
      IF (X(I).LE.X(I-1)) GO TO 33
    1 CONTINUE
C
C        COMPUTE TENSION PARAMETER T
C
      T=ST*FLOAT(N-1)/(X(N)-X(1))
C
C        COMPUTE NEEDED PROGRAM INTEGER VALUES
C
      M=N-1
      N1=N+1
      N2=N1+N
      N3=N2+N
      N4=N3+N
      N5=N4+N
      N6=N5+N
      N7=N6+N
      N8=N7+N
      N9=N8+N
C
C        LOAD MATRIX ROWS 2 THRU N-1
C
      DO 2 I=1,M
      H2=X(I+1)-X(I)
      IF (ST.LE.0.0) YPP(I)=H2/6.
      IF (ST.GT.0.0) YPP(I)=(1./H2-T/SINH(T*H2))/(T*T)
      K=2*N+I
      YPP(K)=YPP(I)
      IF (I.EQ.1) GO TO 2
      K=K+N
      IF (ST.LE.0.0) YPP(K)=(H2+H1)/3.
      IF(ST.GT.0.0)YPP(K)=(T*COSH(T*H1)/SINH(T*H1)-1./H1+T*COSH(T*H2)/SI
     $NH(T*H2)-1./H2)/(T*T)
      K=K+N
      YPP(K)=(Y(I+1)-Y(I))/H2-(Y(I)-Y(I-1))/H1
    2 H1=H2
C
C        LOAD MATRIX ROWS 1 AND N (END CONDITIONS)
C
      YPP(N3)=YPP(N4-1)=1.0
      YPP(N2)=YPP(M)=0.0
C
C      ROW 1
C
      GO TO (3,4,6)OPT(1)
    3 A1=X(2)-X(3)
      A2=X(3)-X(1)
      A3=X(1)-X(2)
      YPP(N4)=2.*(Y(1)*A1+Y(2)*A2+Y(3)*A3)/(A1*X(1)**2+A2*X(2)**2+A3*
     $X(3)**2)
```

```
      GO TO 7
    4 H1=X(2)-X(1)
      YPP(N4)=EC1-(Y(2)-Y(1))/H1
      IF (ST.GT.0.0) GO TO 5
      YPP(N3)=-H1/3.
      YPP(N2)=-H1/6.
      GO TO 7
    5 YPP(N3)=(1./(H1*T)-COSH(T*H1)/SINH(T*H1))/T
      YPP(N2)=(1./SINH(T*H1)-1./(H1*T))/T
      GO TO 7
    6 YPP(N4)=FC1
C
C     ROW N
C
    7 GO TO (8,9,11)OPT(2)
    8 A1=X(N-1)-X(N)
      A2=X(N)-X(N-2)
      A3=X(N-2)-X(N-1)
      YPP(N5-1)=2.*(Y(N-2)*A1+Y(N-1)*A2+Y(N)*A3)/(A1*X(N-2)**2+A2*X(N-1)
     $**2+A3*X(N)**2)
      GO TO 12
    9 H1=X(N)-X(M)
      YPP(N5-1)=ECN-(Y(N)-Y(M))/H1
      IF (ST.GT.0.0) GO TO 10
      YPP(M)=H1/6.
      YPP(N4-1)=H1/3.
      GO TO 12
   10 YPP(M)=(1./(H1*T)-1./SINH(T*H1))/T
      YPP(N4-1)=(COSH(H1*T)/SINH(T*H1)-1./(H1*T))/T
      GO TO 12
   11 YPP(N5-1)=ECN
C
C        SOLVE TRIDIAGONAL MATRIX
C
   12 YPP(N7)=YPP(N3)
      YPP(N8)=YPP(N4)/YPP(N3)
      YPP(N9)=YPP(N2)/YPP(N3)
      DO 13 I=2,N
      I1=I-1
      I2=I-2
      YPP(N7+I1)=YPP(N3+I1)-YPP(I1)*YPP(N9+I2)
      IF (I.EQ.N) GO TO 13
      YPP(N9+I1)=YPP(N2+I1)/YPP(N7+I1)
   13 YPP(N8+I1)=(YPP(N4+I1)-YPP(I1)*YPP(N8+I2))/YPP(N7+I1)
      YPP(N6-1)=YPP(N9-1)
      DO 14 I=1,M
      I1=N6-1-I
      I2=N-I
      I3=I2-1
```

```
   14 YPP(I1)=YPP(N8+I3)-YPP(N9+I3)*YPP(I1+1)
      K=0
      DO 15 I=1,N
      I1=I-1
      YPP(N6+K)=YPP(N5+I1)
   15 K=K+1
      YPP(M)=YPP(N3-2)
C
C        REARRANGE SOLUTION IN YPP ARRAY
C
      DO 16 I=1,N
      I1=N6+I-1
   16 YPP(I)=YPP(I1)
C
      RETURN
   32 IERR=1
      RETURN
   33 IERR=2
      WRITE(6,35)(I,X(I),Y(I),I=1,N)
      RETURN
   35 FORMAT(1H1,5X,33HX IS NOT MONOTONICALLY INCREASING//9X,1HI,13X,1H
     $X,19X,1HY/(5X,I5,5X,E15.6,5X,E15.6))
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      FUNCTION CURV2 (T,N,X,Y,YP,SIGMA)
C
      INTEGER N
      REAL T,X(N),Y(N),YP(N),SIGMA
C
C                             FROM THE SPLINE UNDER TENSION PACKAGE
C                              CODED BY A. K. CLINE AND R. J. RENKA
C                                  DEPARTMENT OF COMPUTER SCIENCES
C                                    UNIVERSITY OF TEXAS AT AUSTIN
C
C THIS FUNCTION INTERPOLATES A CURVE AT A GIVEN POINT
C USING A SPLINE UNDER TENSION. THE SUBROUTINE CURV1 SHOULD
C BE CALLED EARLIER TO DETERMINE CERTAIN NECESSARY
C PARAMETERS.
C
C ON INPUT--
C
C   T CONTAINS A REAL VALUE TO BE MAPPED ONTO THE INTERPO-
C   LATING CURVE.
C
C   N CONTAINS THE NUMBER OF POINTS WHICH WERE SPECIFIED TO
C   DETERMINE THE CURVE.
C
C   X AND Y ARE ARRAYS CONTAINING THE ABSCISSAE AND
C   ORDINATES, RESPECTIVELY, OF THE SPECIFIED POINTS.
```

```
C
C   YP IS AN ARRAY OF SECOND DERIVATIVE VALUES OF THE CURVE
C   AT THE NODES.
C
C AND
C
C   SIGMA CONTAINS THE TENSION FACTOR (ITS SIGN IS IGNORED).
C
C THE PARAMETERS N, X, Y, YP, AND SIGMA SHOULD BE INPUT
C UNALTERED FROM THE OUTPUT OF CURV1.
C
C ON OUTPUT--
C
C   CURV2 CONTAINS THE INTERPOLATED VALUE.
C
C NONE OF THE INPUT PARAMETERS ARE ALTERED.
C
C THIS FUNCTION REFERENCES PACKAGE MODULES INTRVL AND
C SNHCSH.
C
C-----------------------------------------------------------
C
C DETERMINE INTERVAL
C
      IM1 = INTRVL(T,X,N)
      I = IM1+1
C
C DENORMALIZE TENSION FACTOR
C
      SIGMAP = ABS(SIGMA)*FLOAT(N-1)/(X(N)-X(1))
C
C SET UP AND PERFORM INTERPOLATION
C
      DEL1 = T-X(IM1)
      DEL2 = X(I)-T
      DELS = X(I)-X(IM1)
      SUM = (Y(I)*DEL1+Y(IM1)*DEL2)/DELS
      IF (SIGMAP .NE. 0.) GO TO 1
      CURV2 = SUM-DEL1*DEL2*(YP(I)*(DEL1+DELS)+YP(IM1)*
     *        (DEL2+DELS))/(6.*DELS)
      RETURN
    1 DELP1 = SIGMAP*(DEL1+DELS)/2.
      DELP2 = SIGMAP*(DEL2+DELS)/2.
      CALL SNHCSH (SINHM1,DUMMY,SIGMAP*DEL1,-1)
      CALL SNHCSH (SINHM2,DUMMY,SIGMAP*DEL2,-1)
      CALL SNHCSH (SINHMS,DUMMY,SIGMAP*DELS,-1)
      CALL SNHCSH (SINHP1,DUMMY,SIGMAP*DEL1/2.,-1)
      CALL SNHCSH (SINHP2,DUMMY,SIGMAP*DEL2/2.,-1)
      CALL SNHCSH (DUMMY,COSHP1,DELP1,1)
```

```
      CALL SNHCSH (DUMMY,COSHP2,DELP2,1)
      CURV2 = SUM+(YP(I)*(SINHM1*DEL2-DEL1*(2.*(COSHP1+1.)*
     *        SINHP2+SIGMAP*COSHP1*DEL2))+YP(IM1)*(SINHM2*
     *        DEL1-DEL2*(2.*(COSHP2+1.)*SINHP1+SIGMAP*
     *        COSHP2*DEL1)))/(SIGMAP*SIGMAP*DELS*(SINHMS+
     *        SIGMAP*DELS))
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      FUNCTION CURVD (T,N,X,Y,YP,SIGMA)
C
      INTEGER N
      REAL T,X(N),Y(N),YP(N),SIGMA
C
C                       FROM THE SPLINE UNDER TENSION PACKAGE
C                        CODED BY A. K. CLINE AND R. J. RENKA
C                             DEPARTMENT OF COMPUTER SCIENCES
C                               UNIVERSITY OF TEXAS AT AUSTIN
C
C THIS FUNCTION DIFFERENTIATES A CURVE AT A GIVEN POINT
C USING A SPLINE UNDER TENSION. THE SUBROUTINE CURV1 SHOULD
C BE CALLED EARLIER TO DETERMINE CERTAIN NECESSARY
C PARAMETERS.
C
C ON INPUT--
C
C   T CONTAINS A REAL VALUE AT WHICH THE DERIVATIVE IS TO BE
C   DETERMINED.
C
C   N CONTAINS THE NUMBER OF POINTS WHICH WERE SPECIFIED TO
C   DETERMINE THE CURVE.
C
C   X AND Y ARE ARRAYS CONTAINING THE ABSCISSAE AND
C   ORDINATES, RESPECTIVELY, OF THE SPECIFIED POINTS.
C
C   YP IS AN ARRAY OF SECOND DERIVATIVE VALUES OF THE CURVE
C   AT THE NODES.
C
C AND
C
C   SIGMA CONTAINS THE TENSION FACTOR (ITS SIGN IS IGNORED).
C
C THE PARAMETERS N, X, Y, YP, AND SIGMA SHOULD BE INPUT
C UNALTERED FROM THE OUTPUT OF CURV1.
C
C ON OUTPUT--
C
C   CURVD CONTAINS THE DERIVATIVE VALUE.
C
```

```
C NONE OF THE INPUT PARAMETERS ARE ALTERED.
C
C THIS FUNCTION REFERENCES PACKAGE MODULES INTRVL AND
C SNHCSH.
C
C-----------------------------------------------------------
C
C DETERMINE INTERVAL
C
      IM1 = INTRVL(T,X,N)
      I = IM1+1
C
C DENORMALIZE TENSION FACTOR
C
      SIGMAP = ABS(SIGMA)*FLOAT(N-1)/(X(N)-X(1))
C
C SET UP AND PERFORM DIFFERENTIATION
C
      DEL1 = T-X(IM1)
      DEL2 = X(I)-T
      DELS = X(I)-X(IM1)
      SUM = (Y(I)-Y(IM1))/DELS
      IF (SIGMAP .NE. 0.) GO TO 1
      CURVD = SUM+(YP(I)*(2.*DEL1*DEL1-DEL2*(DEL1+DELS))-
     *                YP(IM1)*(2.*DEL2*DEL2-DEL1*(DEL2+DELS)))
     *                /(6.*DELS)
      RETURN
    1 CALL SNHCSH (DUMMY,COSHM1,SIGMAP*DEL1,1)
      CALL SNHCSH (DUMMY,COSHM2,SIGMAP*DEL2,1)
      CALL SNHCSH (SINHMS,DUMMY,SIGMAP*DELS,-1)
      CURVD = SUM+(YP(I)*(DELS*SIGMAP*COSHM1-SINHMS)-
     *          YP(IM1)*(DELS*SIGMAP*COSHM2-SINHMS))/
     *          (SIGMAP*SIGMAP*DELS*(SINHMS+SIGMAP*DELS))
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      FUNCTION CURVI (XL,XU,N,X,Y,YP,SIGMA)
C
      INTEGER N
      REAL XL,XU,X(N),Y(N),YP(N),SIGMA
C
C                             FROM THE SPLINE UNDER TENSION PACKAGE
C                              CODED BY A. K. CLINE AND R. J. RENKA
C                                   DEPARTMENT OF COMPUTER SCIENCES
C                                     UNIVERSITY OF TEXAS AT AUSTIN
C
C THIS FUNCTION INTEGRATES A CURVE SPECIFIED BY A SPLINE
```

```
C UNDER TENSION BETWEEN TWO GIVEN LIMITS. THE SUBROUTINE
C CURV1 SHOULD BE CALLED EARLIER TO DETERMINE NECESSARY
C PARAMETERS.
C
C ON INPUT--
C
C   XL AND XU CONTAIN THE UPPER AND LOWER LIMITS OF INTE-
C   GRATION, RESPECTIVELY. (SL NEED NOT BE LESS THAN OR
C   EQUAL TO XU, CURVI (XL,XU,...) .EQ. -CURVI (XU,XL,...) ).
C
C   N CONTAINS THE NUMBER OF POINTS WHICH WERE SPECIFIED TO
C   DETERMINE THE CURVE.
C
C   X AND Y ARE ARRAYS CONTAINING THE ABSCISSAE AND
C   ORDINATES, RESPECTIVELY, OF THE SPECIFIED POINTS.
C
C   YP IS AN ARRAY FROM SUBROUTINE CURV1 CONTAINING
C   AT THE NODES.
C
C AND
C
C   SIGMA CONTAINS THE TENSION FACTOR (ITS SIGN IS IGNORED).
C
C THE PARAMETERS N, X, Y, YP, AND SIGMA SHOULD BE INPUT
C UNALTERED FROM THE OUTPUT OF CURV1.
C
C ON OUTPUT--
C
C   CURVI CONTAINS THE INTEGRAL VALUE.
C
C NONE OF THE INPUT PARAMETERS ARE ALTERED.
C
C THIS FUNCTION REFERENCES PACKAGE MODULES INTRVL AND
C SNHCSH.
C
C-----------------------------------------------------------
C
C STATEMENT FUNCTION FOR COEFFICIENT ASSOCIATED WITH
C DERIVATIVE TERMS
C
      TERM (CMM1,CMM2,T) = (CMM1-CMM2-SIGMAP*T*SS)/(SIGMAP*
     *                      SIGMAP*SIGMAP*(SS+SIGMAP*DELS))
C
C DENORMALIZE TENSION FACTOR
C
      SIGMAP = ABS(SIGMA)*FLOAT(N-1)/(X(N)-X(1))
C
C DETERMINE ACTUAL UPPER AND LOWER BOUNDS
C
      XXL = XL
      XXU = XU
```

```
      SSIGN = 1.
      IF (XL .LT. XU) GO TO 1
      XXL = XU
      XXU = XL
      SSIGN = -1.
      IF (XL .GT. XU) GO TO 1
C
C RETURN ZERO IF XL .EQ. XU
C
      CURVI = 0.
      RETURN
C
C SEARCH FOR PROPER INTERVALS
C
    1 ILM1 = INTRVL (XXL,X,N)
      IL = ILM1+1
      IUM1 = INTRVL (XXU,X,N)
      IU = IUM1+1
      IF (IL .EQ. IU) GO TO 8
C
C INTEGRATE FROM XXL TO X(IL)
C
      SUM = 0.
      IF (XXL .EQ. X(IL)) GO TO 3
      DEL1 = XXL-X(ILM1)
      DEL2 = X(IL)-XXL
      DELS = X(IL)-X(ILM1)
      T1 = (DEL1+DELS)*DEL2/(2.*DELS)
      T2 = DEL2*DEL2/(2.*DELS)
      SUM = T1*Y(IL)+T2*Y(ILM1)
      IF (SIGMA .EQ. 0.) GO TO 2
      CALL SNHCSH (DUMMY,C1,SIGMAP*DEL1,2)
      CALL SNHCSH (DUMMY,C2,SIGMAP*DEL2,2)
      CALL SNHCSH (SS,CS,SIGMAP*DELS,3)
      SUM = SUM+TERM(CS,C1,T1)*YP(IL)
     *          +TERM(C2,0.,T2)*YP(ILM1)
      GO TO 3
    2 SUM = SUM-T1*T1*DELS*YP(IL)/6.
     *          -T2*(DEL1*(DEL2+DELS)+DELS*DELS)*YP(ILM1)/12.
C
C INTEGRATE OVER INTERIOR INTERVALS
C
    3 IF (IU-IL .EQ. 1) GO TO 6
      ILP1 = IL+1
      DO 5 I = ILP1,IUM1
        DELS = X(I)-X(I-1)
        SUM = SUM+(Y(I)+Y(I-1))*DELS/2.
        IF (SIGMA .EQ. 0.) GO TO 4
        CALL SNHCSH (SS,CS,SIGMAP*DELS,3)
```

```
      SUM = SUM+(YP(I)+YP(I-1))*(CS-SS*SIGMAP*DELS/2.)/
     *            (SIGMAP*SIGMAP*SIGMAP*(SS+SIGMAP*DELS))
      GO TO 5
    4 SUM = SUM-(YP(I)+YP(I-1))*DELS*DELS*DELS/24.
    5 CONTINUE
C
C INTEGRATE FROM X(IU-1) TO XXU
C
    6 IF (XXU .EQ. X(IUM1)) GO TO 10
      DEL1 = XXU-X(IUM1)
      DEL2 = X(IU)-XXU
      DELS = X(IU)-X(IUM1)
      T1 = DEL1*DEL1/(2.*DELS)
      T2 = (DEL2+DELS)*DEL1/(2.*DELS)
      SUM = SUM+T1*Y(IU)+T2*Y(IUM1)
      IF (SIGMA .EQ. 0.) GO TO 7
      CALL SNHCSH (DUMMY,C1,SIGMAP*DEL1,2)
      CALL SNHCSH (DUMMY,C2,SIGMAP*DEL2,2)
      CALL SNHCSH (SS,CS,SIGMAP*DELS,3)
      SUM = SUM+TERM(C1,0.,T1)*YP(IU)
     *          +TERM(CS,C2,T2)*YP(IUM1)
      GO TO 10
    7 SUM = SUM-T1*(DEL2*(DEL1+DELS)+DELS*DELS)*YP(IU)/12.
     *          -T2*T2*DELS*YP(IUM1)/6.
      GO TO 10
C
C INTEGRATE FROM XXL TO XXU
C
    8 DELU1 = XXU-X(IUM1)
      DELU2 = X(IU)-XXU
      DELL1 = XXL-X(IUM1)
      DELL2 = X(IU)-XXL
      DELS = X(IU)-X(IUM1)
      DELI = XXU-XXL
      T1 = (DELU1+DELL1)*DELI/(2.*DELS)
      T2 = (DELU2+DELL2)*DELI/(2.*DELS)
      SUM = T1*Y(IU)+T2*Y(IUM1)
      IF (SIGMA .EQ. 0.) GO TO 9
      CALL SNHCSH (DUMMY,CU1,SIGMAP*DELU1,2)
      CALL SNHCSH (DUMMY,CU2,SIGMAP*DELU2,2)
      CALL SNHCSH (DUMMY,CL1,SIGMAP*DELL1,2)
      CALL SNHCSH (DUMMY,CL2,SIGMAP*DELL2,2)
      CALL SNHCSH (SS,DUMMY,SIGMAP*DELS,-1)
      SUM = SUM+TERM(CU1,CL1,T1)*YP(IU)
     *          +TERM(CL2,CU2,T2)*YP(IUM1)
      GO TO 10
    9 SUM = SUM-T1*(DELU2*(DELS+DELU1)+DELL2*(DELS+DELL1))*
     *              YP(IU)/12.
     *          -T2*(DELL1*(DELS+DELL2)+DELU1*(DELS+DELU2))*
     *              YP(IUM1)/12.
C
```

```
C CORRECT SIGN AND RETURN
C
   10 CURVI = SSIGN*SUM
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      FUNCTION INTRVL (T,X,N)
C
      INTEGER N
      REAL T,X(N)
C
C                                 FROM THE SPLINE UNDER TENSION PACKAGE
C                                  CODED BY A. K. CLINE AND R. J. RENKA
C                                       DEPARTMENT OF COMPUTER SCIENCES
C                                         UNIVERSITY OF TEXAS AT AUSTIN
C
C THIS FUNCTION DETERMINES THE INDEX OF THE INTERVAL
C (DETERMINED BY A GIVEN INCREASING SEQUENCE) IN WHICH
C A GIVEN VALUE LIES.
C
C ON INPUT--
C
C   T IS THE GIVEN VALUE.
C
C   X IS A VECTOR OF STRICTLY INCREASING VALUES.
C
C AND
C
C   N IS THE LENGTH OF X (N .GE. 2).
C
C ON OUTPUT--
C
C   INTRVL RETURNS AN INTEGER I SUCH THAT
C
C          I = 1       IF              T .LE. X(2)   ,
C          I = N-1     IF X(N-1) .LE. T              ,
C          OTHERWISE      X(I)   .LE. T .LE. X(I+1),
C
C NONE OF THE INPUT PARAMETERS ARE ALTERED.
C
C-----------------------------------------------------------
C
      TT = T
      IF (TT .LE. X(2)) GO TO 4
      IF (TT .GE. X(N-1)) GO TO 5
      IL = 2
      IH = N-1
```

```
C
C LINEAR INTERPOLATION
C
    1 I = IL+IFIX(FLOAT(IH-IL)*(TT-X(IL))/(X(IH)-X(IL)))
      IF (TT .LT. X(I)) GO TO 2
      IF (TT .LE. X(I+1)) GO TO 3
C
C TOO HIGH
C
      IL = I+1
      GO TO 1
C
C TOO LOW
C
    2 IH = I
      GO TO 1
    3 INTRVL = I
      RETURN
C
C LEFT END
C
    4 INTRVL = 1
      RETURN
C
C RIGHT END
C
    5 INTRVL = N-1
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE SNHCSH (SINHM,COSHM,X,ISW)
C
      INTEGER ISW
      REAL SINHM,COSHM,X
C
C
C                        FROM THE SPLINE UNDER TENSION PACKAGE
C                         CODED BY A. K. CLINE AND R. J. RENKA
C                             DEPARTMENT OF COMPUTER SCIENCES
C                               UNIVERSITY OF TEXAS AT AUSTIN
C
C THIS SUBROUTINE RETURNS APPROXIMATIONS TO
C       SINHM(X) = SINH(X)-X
C       COSHM(X) = COSH(X)-1
C AND
C       COSHMM(X) = COSH(X)-1-X*X/2
C WITH RELATIVE ERROR LESS THAN 3.42E-14
C
C ON INPUT--
C
```

```
C    X CONTAINS THE VALUE OF THE INDEPENDENT VARIABLE.
C
C    ISW INDICATES THE FUNCTION DESIRED
C            = -1 IF ONLY SINHM IS DESIRED,
C            =  0 IF BOTH SINHM AND COSHM ARE DESIRED,
C            =  1 IF ONLY COSHM IS DESIRED,
C            =  2 IF ONLY COSHMM IS DESIRED,
C            =  3 IF BOTH SINHM AND COSHMM ARE DESIRED.
C
C ON OUTPUT--
C
C    SINHM CONTAINS THE VALUE OF SINHM(X) IF ISW .LE. 0 OR
C    ISW .EQ. 3 (SINHM IS UNALTERED IF ISW .EQ.1 OR ISW .EQ.
C    2).
C
C    COSHM CONTAINS THE VALUE OF COSHM(X) IF ISW .EQ. 0 OR
C    ISW .EQ. 1 AND CONTAINS THE VALUE OF COSHMM(X) IF ISW
C    .GE. 2 (COSHM IS UNALTERED IF ISW .EQ. -1).
C
C AND
C
C    X AND ISW ARE UNALTERED.
C
C-----------------------------------------------------------
C
      DATA SP4/4.50217693381333E-08/,
     *     SP3/8.95278544216390E-06/,
     *     SP2/8.72048976791502E-04/,
     *     SP1/4.36314556981690E-02/,
     *     SQ1/-6.36854430175110E-03/
      DATA CP4/1.78419567490190E-07/,
     *     CP3/2.87277229799044E-05/,
     *     CP2/2.15151519902028E-03/,
     *     CP1/7.58181822756256E-02/,
     *     CQ1/-7.51515105679867E-03/
      DATA ZP3/5.59297116264720E-07/,
     *     ZP2/1.77943488030894E-04/,
     *     ZP1/1.69800461894792E-02/,
     *     ZQ4/1.33412535492375E-09/,
     *     ZQ3/-5.80858944138663E-07/,
     *     ZQ2/1.27814964403863E-04/,
     *     ZQ1/-1.63532871439181E-02/
      XX = X
      AX = ABS(XX)
      XS = XX*XX
      IF ((AX .GE. 2.70) .OR. (AX .GE. 1.15 .AND.
     *     ISW .NE. 2)) EXPX = EXP(AX)
C
C SINHM APPROXIMATION
```

```
C
      IF (ISW .EQ. 1 .OR. ISW .EQ. 2) GO TO 2
      IF (AX .GE. 1.15) GO TO 1
      SINHM = (((((SP4*XS+SP3)*XS+SP2)*XS+SP1)*XS+1.)*XS*XX)
     *        /((SQ1*XS+1.)*6.)
      GO TO 2
    1 SINHM = -(((1./EXPX+AX)+AX)-EXPX)/2.
      IF (XX .LT. 0.) SINHM = -SINHM
C
C COSHM APPROXIMATION
C
    2 IF (ISW .NE. 0 .AND. ISW .NE. 1) GO TO 4
      IF (AX .GE. 1.15) GO TO 3
      COSHM = (((((CP4*XS+CP3)*XS+CP2)*XS+CP1)*XS+1.)*XS)
     *        /((CQ1*XS+1.)*2.)
      GO TO 4
    3 COSHM = ((1./EXPX-2.)+EXPX)/2.
C
C COSHMM APPROXIMATION
C
    4 IF (ISW .LE. 1) RETURN
      IF (AX .GE. 2.70) GO TO 5
      COSHM = ((((ZP3*XS+ZP2)*XS+ZP1)*XS+1.)*XS*XS)/(((((ZQ4
     *        *XS+ZQ3)*XS+ZQ2)*XS+ZQ1)*XS+1.)*24.)
      RETURN
    5 COSHM = (((1./EXPX-2.)-XS)+EXPX)/2.
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THE SUBROUTINE AVERAGES THE TOP AND BOTTOM WALL VALUES OF DYJ
C
      SUBROUTINE AVGDYJ(NJ,XJ,DYJ)
      DIMENSION XJ(NJ),DYJ(NJ)
      N=NJ/2
      IF((2*N).NE.NJ)GO TO 20
      DO 10 I=1,N
      SIGN=1.0
      IF(DYJ(I).LT.0.0)SIGN=-1.0
      DYA=(ABS(DYJ(I))+ABS(DYJ(I+N)))/2.0
      DYJ(I)=DYA*SIGN
      DYJ(I+N)=-DYJ(I)
   10 CONTINUE
      WRITE(6,99)
      DO 15 I=1,N
      WRITE(6,98)I,XJ(I),DYJ(I),XJ(I+N),DYJ(I+N)
   15 CONTINUE
      RETURN
   20 WRITE(6,97)
      RETURN
```

```
 97 FORMAT("1"///* REQUEST FOR AVERAGING DENIED*/
   $*      NJ IS AN ODD NUMBER OF JACKS*/)
 98 FORMAT(2XI2,4(2XG12.6))
 99 FORMAT("1"//* RESULTS OF AVERAGING DYJ OF TOP AND*
   $* BOTTOM WALLS*//3X*I*7X*XJ(TOP)*6X*DYJ(TOP)*7X*XJ(BOT)*
   $6X*DYJ(BOT)*/)
    END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THIS SUBROUTINE WRITES THE PREDICTED RESULTS ON TO TAPE6
C
      SUBROUTINE NEWYJ(IRLAX,NRDY,RDY)
      REAL M,MSET
      INTEGER TEST,RUN,POINT
      COMMON/JACK/NJ,NJT,NJB,XJ(100),YJ(100),UJ(100),VJ(100),
     $UJN(100),VJN(100),YJN(100),DYJ(100)
      COMMON/PARAM/HEAD(8),TEST,RUN,POINT,M,RN,ALPHA,TEMP,RLAX(10),
     $NRLAX,PT,PINF,UINF,IAVG,IS(4),IOPT(4),EC(4),GAMMA,MSET
      DIMENSION RDY(NRDY)
      DO 10 I=1,NJ
      RDY(I)=RLAX(IRLAX)*DYJ(I)
      YJN(I)=YJ(I)+RDY(I)
   10 CONTINUE
      RMSDYTC=RMS(NJT,RDY(1))
      RMSDYBC=RMS(NJB,RDY(NJT+1))
      WRITE(6,983)RLAX(IRLAX)
      WRITE(6,984)(HEAD(I),I=1,8)
      WRITE(6,985)TEST,RUN,POINT
      WRITE(6,986)MSET,RN,ALPHA,TEMP,PT,PINF,UINF
      WRITE(6,987)
C     WRITE(7,991)TEST,RUN,POINT,RLAX(IRLAX),MSET
      DO 915 I=1,NJ
      WRITE(6,988)I,XJ(I),YJ(I),YJN(I),RDY(I)
C     WRITE(7,990)XJ(I),YJ(I),RDY(I),YJN(I)
  915 CONTINUE
      WRITE(6,989)RMSDYTC,RMSDYBC
      RETURN
  983 FORMAT("1"//*     UNSCALED WALL CORRECTIONS FOR A RELAXATION*
     $* FACTOR=*F10.5)
  984 FORMAT(//3X8A10)
  985 FORMAT(3X,*USING RESULTS FROM TEST *,I4,*  RUN *,I4,*  POINT *,I4)
  986 FORMAT(//,3X,*TUNNEL PARAMETERS ARE*,/,10X,*M=*,F8.4,4X,*RN=*,G9.4
     $,4X,*ALPHA=*,F8.3,3X,*TEMP(DEG R)=*,F8.3,/,10X,*PT=*,G9.4,3X,
     $*PINF=*,G9.4,1X,*UINF=*,F11.5)
  987 FORMAT(//4X*I*6X*XJ*11X*YJ*11X*YJN*11X*DYJ*/)
  988 FORMAT(2XI3,4(2XG12.6))
  989 FORMAT(//33X*RMS(DYJ(TOP))=*G12.6/
     $         33X*RMS(DYJ(BOT))=*G12.6)
```

```
990 FORMAT(4F10.5)
991 FORMAT(* TEST=*I3*  RUN=*I3*  POINT=*I3*  RLAX=*F6.4*  MSET=*
   %F10.5)
    END
```

APPENDIX C

SAMPLE CASE

INPUT DATA LISTING

```
6X19 INCH TRANSONIC TUNNEL FLEXIBLE WALL TEST
   39   31   35
 3.0728     0.          508.47      20.616      13.914      .76700
   12   12   19    1    19    1     2    0     0    0     0    1    1
    2    1    2    1
0.          0.          0.          0.
 .25000      1.0000
-29.00000    9.50000
-24.00000    9.49765
-20.00000    9.49739
-16.00000    9.50130
-12.00000    9.50918
 -8.00000    9.51911
 -4.00000    9.55012
  0.00000    9.57337
  4.00000    9.54136
  8.00000    9.50557
 12.00000    9.49375
 16.00000    9.49179
-29.00000   -9.50000
-24.00000   -9.49765
-20.00000   -9.49739
-16.00000   -9.50130
-12.00000   -9.50918
 -8.00000   -9.51911
 -4.00000   -9.55012
  0.00000   -9.57337
  4.00000   -9.54136
  8.00000   -9.50557
 12.00000   -9.49375
 16.00000   -9.49179
-26.00000    9.50000    -.00900
-18.00000    9.50000    -.00340
-14.00000    9.50000    -.00300
-10.00000    9.50000    -.00830
 -7.00000    9.50000    -.00440
 -6.00000    9.50000     .00110
 -5.00000    9.50000     .00020
 -3.00000    9.50000    -.03590
 -2.00000    9.50000    -.06440
 -1.00000    9.50000    -.09820
  1.00000    9.50000    -.08770
  2.00000    9.50000    -.05520
```

3.00000	9.50000	-.02040
5.00000	9.50000	.00300
6.00000	9.50000	-.00480
7.00000	9.50000	-.00780
10.00000	9.50000	.00090
14.00000	9.50000	.00610
18.00000	9.50000	.00470
18.00000	0.00000	.00470
18.00000	-9.50000	.00470
14.00000	-9.50000	.00610
10.00000	-9.50000	.00090
7.00000	-9.50000	-.00780
6.00000	-9.50000	-.00480
5.00000	-9.50000	.00300
3.00000	-9.50000	-.02040
2.00000	-9.50000	-.05520
1.00000	-9.50000	-.08770
-1.00000	-9.50000	-.09820
-2.00000	-9.50000	-.06440
-3.00000	-9.50000	-.03590
-5.00000	-9.50000	.00020
-6.00000	-9.50000	.00110
-7.00000	-9.50000	-.00440
-10.00000	-9.50000	-.00830
-14.00000	-9.50000	-.00300
-18.00000	-9.50000	-.00340
-26.00000	-9.50000	-.00900
-26.00000	0.00000	-.00900

M,INF= .770943 UINF= 805.542

NEW JACK PERTURBATION VELOCITY CALCULATIONS (SCALED)

I	XJ	YJ	UJ	UJN	VJ	VJN	DVJ	DUJ
1	-29.0000	6.0506	2.03	2.03	-.2802E-14	-.2802E-14	0.	0.
2	-24.0000	6.0491	1.18	-1.99	-.2428	-.9313E-01	.1497	-3.171
3	-20.0000	6.0489	.74	-2.15	.1943	.7926E-01	-.1151	-2.893
4	-16.0000	6.0514	.41	-2.21	.8707	.7597	-.1111	-2.620
5	-12.0000	6.0564	.90	-1.96	.8623	1.722	.8594	-2.856
6	-8.0000	6.0628	1.31	-2.62	2.542	2.012	-.5299	-3.925
7	-4.0000	6.0825	2.17	1.04	4.754	8.084	3.329	-1.132
8	0.0000	6.0973	17.21	5.97	-.6386	-1.734	-1.095	-11.24
9	4.0000	6.0769	.02	.49	-5.611	-7.729	-2.118	.4667
10	8.0000	6.0541	1.03	-3.12	-3.022	-3.818	-.7957	-4.150
11	12.0000	6.0466	-.83	-2.57	-.6345	-2.634	-2.000	-1.734
12	16.0000	6.0454	-.98	-1.93	.2564	-1.455	-1.711	-.9525
13	-29.0000	-6.0506	2.03	2.03	.2802E-14	.2802E-14	0.	0.
14	-24.0000	-6.0491	1.18	-1.99	.2428	.9313E-01	-.1497	-3.171
15	-20.0000	-6.0489	.74	-2.15	-.1943	-.7926E-01	.1151	-2.893
16	-16.0000	-6.0514	.41	-2.21	-.8707	-.7597	.1111	-2.620
17	-12.0000	-6.0564	.90	-1.96	-.8623	-1.722	-.8594	-2.856
18	-8.0000	-6.0628	1.31	-2.62	-2.542	-2.012	.5299	-3.925
19	-4.0000	-6.0825	2.17	1.04	-4.754	-8.084	-3.329	-1.132
20	0.0000	-6.0973	17.21	5.97	.6386	1.734	1.095	-11.24
21	4.0000	-6.0769	.02	.49	5.611	7.729	2.118	.4667
22	8.0000	-6.0541	1.03	-3.12	3.022	3.818	.7957	-4.150
23	12.0000	-6.0466	-.83	-2.57	.6345	2.634	2.000	-1.734
24	16.0000	-6.0454	-.98	-1.93	-.2564	1.455	1.711	-.9525

50 PREDICTED WALL CORRECTION (SCALED)

I	XJ	YJ	UJ	UJN	VJ	VJN	THETA	THETAN	DYJ
1	-29.0000	6.0506	807.57	807.57	-.2802E-14	-.2802E-14	-.3469E-17	-.3469E-17	0.
2	-24.0000	6.0491	806.72	803.55	-.2428	-.9313E-01	-.3010E-03	-.1159E-03	.7324E-03
3	-20.0000	6.0489	806.28	803.39	.1943	.7926E-01	.2410E-03	.9865E-04	.9689E-03
4	-16.0000	6.0514	805.96	803.34	.8707	.7597	.1080E-02	.9457E-03	-.2384E-03
5	-12.0000	6.0564	806.44	803.58	.8623	1.722	.1069E-02	.2142E-02	.2564E-02
6	-8.0000	6.0628	806.85	802.93	2.542	2.012	.3150E-02	.2506E-02	.2098E-02
7	-4.0000	6.0825	807.72	806.58	4.754	8.084	.5886E-02	.1002E-01	.9879E-02
8	0.0000	6.0973	822.75	811.51	-.6386	-1.734	-.7762E-03	-.2136E-02	.1734E-01
9	4.0000	6.0769	805.57	806.03	-5.611	-7.729	-.6966E-02	-.9589E-02	.6981E-02
10	8.0000	6.0541	806.57	802.42	-3.022	-3.818	-.3747E-02	-.4758E-02	.2614E-03
11	12.0000	6.0466	804.71	802.98	-.6345	-2.634	-.7885E-03	-.3280E-02	-.6330E-02
12	16.0000	6.0454	804.57	803.61	.2564	-1.455	.3186E-03	-.1811E-02	-.1653E-01
13	-29.0000	-6.0506	807.57	807.57	.2802E-14	.2802E-14	.3469E-17	.3469E-17	0.
14	-24.0000	-6.0491	806.72	803.55	.2428	.9313E-01	.3010E-03	.1159E-03	-.7324E-03
15	-20.0000	-6.0489	806.28	803.39	-.1943	-.7926E-01	-.2410E-03	-.9865E-04	-.9689E-03
16	-16.0000	-6.0514	805.96	803.34	-.8707	-.7597	-.1080E-02	-.9457E-03	.2384E-03
17	-12.0000	-6.0564	806.44	803.58	-.8623	-1.722	-.1069E-02	-.2142E-02	-.2564E-02
18	-8.0000	-6.0628	806.85	802.93	-2.542	-2.012	-.3150E-02	-.2506E-02	-.2098E-02
19	-4.0000	-6.0825	807.72	806.58	-4.754	-8.084	-.5886E-02	-.1002E-01	-.9879E-02
20	0.0000	-6.0973	822.75	811.51	.6386	1.734	.7762E-03	.2136E-02	-.1734E-01
21	4.0000	-6.0769	805.57	806.03	5.611	7.729	.6966E-02	.9589E-02	-.6981E-02
22	8.0000	-6.0541	806.57	802.42	3.022	3.818	.3747E-02	.4758E-02	-.2614E-03
23	12.0000	-6.0466	804.71	802.98	.6345	2.634	.7885E-03	.3280E-02	.6330E-02
24	16.0000	-6.0454	804.57	803.61	-.2564	1.455	-.3186E-03	.1811E-02	.1653E-01

RMS(DYJ(TOP))= .802531E-02 RMS(DYJ(BOT))= .802531E-02

UNSCALED WALL CORRECTIONS FOR A RELAXATION FACTOR= .25000

6X19 INCH TRANSONIC TUNNEL FLEXIBLE WALL TEST
USING RESULTS FROM TEST 39 RUN 31 POINT 35

TUNNEL PARAMETERS ARE
M= .7670 RN=3.073 ALPHA= 0.000 TEMP(DEG R)= 508.472
PT=20.62 PINF=13.91 UINF= 805.54225

I	XJ	YJ	YJN	DYJ
1	-29.0000	9.50000	9.50000	0.
2	-24.0000	9.49765	9.49794	.285355E-03
3	-20.0000	9.49739	9.49777	.377491E-03
4	-16.0000	9.50130	9.50121	-.928892E-04
5	-12.0000	9.50918	9.51018	.999132E-03
6	-8.00000	9.51911	9.51993	.817311E-03
7	-4.00000	9.55012	9.55397	.384903E-02
8	0.	9.57337	9.58013	.675586E-02
9	4.00000	9.54136	9.54408	.271998E-02
10	8.00000	9.50557	9.50567	.101828E-03
11	12.0000	9.49375	9.49128	-.246630E-02
12	16.0000	9.49179	9.48535	-.644039E-02
13	-29.0000	-9.50000	-9.50000	0.
14	-24.0000	-9.49765	-9.49794	-.285355E-03
15	-20.0000	-9.49739	-9.49777	-.377491E-03
16	-16.0000	-9.50130	-9.50121	.928892E-04
17	-12.0000	-9.50918	-9.51018	-.999132E-03
18	-8.00000	-9.51911	-9.51993	-.817311E-03
19	-4.00000	-9.55012	-9.55397	-.384903E-02
20	0.	-9.57337	-9.58013	-.675586E-02
21	4.00000	-9.54136	-9.54408	-.271998E-02
22	8.00000	-9.50557	-9.50567	-.101828E-03
23	12.0000	-9.49375	-9.49128	.246630E-02
24	16.0000	-9.49179	-9.48535	.644039E-02

RMS(DYJ(TOP))= .312684E-02
RMS(DYJ(BOT))= .312684E-02

UNSCALED WALL CORRECTIONS FOR A RELAXATION FACTOR= 1.00000

6X19 INCH TRANSONIC TUNNEL FLEXIBLE WALL TEST
USING RESULTS FROM TEST 39 RUN 31 POINT 35

TUNNEL PARAMETERS ARE
M= .7670 RN=3.073 ALPHA= 0.000 TEMP(DEG R)= 508.472
PT=20.62 PINF=13.91 UINF= 805.54225

I	XJ	YJ	YJN	DYJ
1	-29.0000	9.50000	9.50000	0.
2	-24.0000	9.49765	9.49879	.114142E-02
3	-20.0000	9.49739	9.49890	.150996E-02
4	-16.0000	9.50130	9.50093	-.371557E-03
5	-12.0000	9.50918	9.51318	.399653E-02
6	-8.00000	9.51911	9.52238	.326924E-02
7	-4.00000	9.55012	9.56552	.153961E-01
8	0.	9.57337	9.60039	.270234E-01
9	4.00000	9.54136	9.55224	.108799E-01
10	8.00000	9.50557	9.50598	.407313E-03
11	12.0000	9.49375	9.48388	-.986519E-02
12	16.0000	9.49179	9.46603	-.257616E-01
13	-29.0000	-9.50000	-9.50000	0.
14	-24.0000	-9.49765	-9.49879	-.114142E-02
15	-20.0000	-9.49739	-9.49890	-.150996E-02
16	-16.0000	-9.50130	-9.50093	.371557E-03
17	-12.0000	-9.50918	-9.51318	-.399653E-02
18	-8.00000	-9.51911	-9.52238	-.326924E-02
19	-4.00000	-9.55012	-9.56552	-.153961E-01
20	0.	-9.57337	-9.60039	-.270234E-01
21	4.00000	-9.54136	-9.55224	-.108799E-01
22	8.00000	-9.50557	-9.50598	-.407313E-03
23	12.0000	-9.49375	-9.48388	.986519E-02
24	16.0000	-9.49179	-9.46603	.257616E-01

RMS(DYJ(TOP))= .125074E-01
RMS(DYJ(BOT))= .125074E-01

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-84648		

4. Title and Subtitle	5. Report Date
FLEXWAL: A Computer Program for Predicting the Wall Modifications for Two-Dimensional, Solid, Adaptive-Wall Wind Tunnels	November 1983
	6. Performing Organization Code 505-31-23-06
7. Author(s)	8. Performing Organization Report No.
Joel L. Everhart	
	10. Work Unit No.
9. Performing Organization Name and Address	
NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

This work was initiated while the author was a graduate research scholar assistant in The George Washington University Joint Institute for the Advancement of Flight Sciences.

16. Abstract

A program called FLEXWAL for calculating wall modifications for solid, adaptive-wall wind tunnels is presented. The method used is the iterative technique of NASA TP-2081 and is applicable to subsonic and transonic test conditions. The program usage, program listing, and a sample case are given.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Wind Tunnels Wall Interference Adaptive Wind-Tunnel Walls Transonic Flow Cauchy Integral Formula	~~FEDD Distribution~~ Subject Category 02

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	53	

Available: NASA's Industrial Applications Centers